RL-TR-93-154
Final Technical Report
July 1993

GENERALIZED MULTICHANNEL SIGNAL DETECTION

Kaman Sciences Corporation

Robert L. Vienneau, David Dekkers, and Sue Eilers

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-93-154 has been reviewed and is approved for publication.

APPROVED: James H. Michels

JAMES H. MICHELS
Project Engineer

FOR THE COMMANDER:

JAMES W. YOUNGBERG, Lt Col, USAF
Deputy Director
Surveillance & Photonics Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL (OCTM) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 1993	Final Nov 92 - Nov 93

4. TITLE AND SUBTITLE
GENERALIZED MULTICHANNEL SIGNAL DETECTION

5. FUNDING NUMBERS
C - F30602-92-C-0033
PE - 62702F
PR - 4506
TA - 17
WU - P7

6. AUTHOR(S)
Robert L. Vienneau, David Dekkers, and Sue Eilers

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Kaman Sciences Corporation
258 Genesee Street
Utica NY 13502-4627

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Rome Laboratory (OCTM)
26 Electronic Pky
Griffiss AFB NY 13441-4514

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
RL-TR-93-154

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: James H. Michels/OCTM/(315) 330-4431

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The Multichannel Signal Processing Simulation System (MSPSS) provides:

o A synthesis procedure for simulating multichannel Autoregressive (AR) processes in which intertemporal and interchannel correlations are controlled by specifying signal characteristics such as the one-lag temporal and cross-channel correlation parameters.

o A capability for analyzing the performance of various multichannel estimation algorithms.

o A multichannel signal detection algorithm based on a generalized likelihood ratio using an innovations approach.

Under this effort the MSPSS was enhanced to provide synthesis and analysis of more general processes including multipath and feedback processes, non-Gaussian Spherically Invariant Random Processes, and processes with nonconstrained quadrature components.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Multichannel Time Series, Multipath Processes, Model-Based Detection, Spherically Invariant Random Processes	44
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

1. OVERVIEW 3
   1.1 Background 3
   1.2 Objective 4
   1.3 Tasks Performed 5
      1.3.1 Task 1: Process Synthesis 5
         1.3.1.1 Unconstrained Quadrature Components 5
         1.3.1.2 Multipath Processes 5
         1.3.1.3 Spherically Invariant Random Processes 5
         1.3.1.4 Processes with Abrupt Parameter Changes 6
      1.3.2 Task 2: Algorithm Implementation 6
      1.3.3 Task 3: Likelihood Ratio Implementation 6
         1.3.3.1 Unconstrained Quadrature Likelihood Ratio 6
         1.3.3.2 SIRP Likelihood Ratio 7
      1.3.4 Task 4: Diagnostics Implementation 7
   1.4 Summary of Results 7

2. THE MULTICHANNEL SIGNAL PROCESSING SIMULATION SYSTEM 8
   2.1 The Multichannel Signal Detection Problem 8
      2.1.1 An Innovations Approach 8
      2.1.2 Shaping Functions and the Yule-Walker Equations 9
      2.1.3 The Signal Detection Algorithm 10
   2.2 Using the MSPSS 12
   2.3 Adding Capabilities to the Old System 12
   2.4 Adding Capabilities to the New System 13
      2.4.1 Source Modules 14
      2.4.2 Knowledge Base 14
      2.4.3 Sequences and Menus 15

3. NEW CAPABILITIES 16
   3.1 Process Synthesis 16
      3.1.1 Quadrature Synthesis 16
         3.1.1.1 White Noise 17
         3.1.1.2 AR Process Generation 18
         3.1.1.3 Implementation Notes 19
      3.1.2 Conversion to Complex Form 19
      3.1.3 SIRP Synthesis 20
         3.1.3.1 Deterministic Signal 21

3.1.3.2 Noise 21
3.1.3.2.1 Gaussian Noise 21
3.1.3.2.2 SIRP Noise 22
3.1.3.3 AR Process 23
3.1.3.4 Multipath Processes 23
3.1.3.5 Program Input 24
3.1.4 The ASCII File Minisystem 24
3.1.5 Abrupt Parameter Changes 24
3.2 Algorithms 25
3.3 Likelihood Ratios 25
3.3.1 The Unconstrained Quadrature Loglikelihood Statistic 25
3.3.2 The SIRP Loglikelihood Statistic 26
3.3.2.1 Background 27
3.3.2.2 Detailed Specification 28
3.4 Diagnostics 29

4. ANALYSIS SEQUENCES 31
4.1 AR Signal Sequences 31
4.2 Multipath Sequences 32
4.3 Deterministic Signal Sequences 33
4.4 Unconstrained Quadrature Sequences 34

5. REFERENCES 35

Appendix A: The Single-Channel Minisystem 36

LIST OF TABLES

Table A-1: Files in the Minisystem 37

1. OVERVIEW

This document is the final report for Generalized Multichannel Signal Detection, an effort conducted by Kaman Sciences Corporation (KSC) under Rome Laboratory's Broad Agency Announcement (BAA) number 90-04. This effort was performed for .

Dr. James H. Michels
Rome Laboratory
RL/OCTM
Griffiss AFB, NY 13441
(315) 330-4432

1.1 Background

Over the last few years, the Surveillance and Photonics Directorate of Rome Laboratory has been investigating an innovative approach in signal processing. This model-based innovations approach has been described in a series of technical reports ([Michels 89], [Michels 90a], [Michels 90b], and [Michels 91]). This investigation has developed:

- A synthesis procedure for simulating multichannel Autoregressive (AR) processes in which intertemporal and interchannel correlations are controlled by specifying signal characteristics such as the one-lag temporal and cross-channel correlation parameters.
- A capability for analyzing the performance of various multichannel estimation algorithms.
- A multichannel signal detection algorithm based on a generalized likelihood ratio using an innovations approach.

Kaman Sciences has designed a software system to support these investigations. This system, the Multichannel Signal Processing Simulation System (MSPSS), allows the user to:

- Synthesize multichannel signals, interference (clutter), and noise as described by certain Autoregressive (AR) models
- Perform certain diagnostic functions on the synthesized signals, such as plotting, calculating certain statistics, and examining Fourier transforms and correlation functions.
- Calculate a loglikelihood statistic and examine its performance in terms of false alarm and detection probabilities.

The MSPSS has two main subsystems. The original subsystem is a collection of stand-alone programs, each of which are called from a variety of menus. These programs pass data between one another in system files. The 'old system' is very flexible in the analysis that it supports in that the user is not limited to executing the programs in a predefined order. It also provides a wide range of functionality. It is quite inconvenient, however, when the user wants to perform repetitive analyses. The user interacts with the system for a short while to execute a program, waits a considerable time for the program to complete, then must interact with the program again, and so on. A batch capability in which the user can define all inputs at once would be much more convenient.

The 'new subsystem' was developed to fill this need ([Kaman 91a] and [Kaman 92b]). The 'new system' was designed and implemented under a Graphical User Interface-based fourth generation system. Kaman Sciences originally designed this GUI-based system, the User Front-end Interface (UFI) as part of the Multisensor Interface Algorithm (MAX) system,

a program developed for RL/OCTS. The UFI combines code generation capabilities with windowing, icons, and a mouse-driven interface. The UFI is general purpose, but the MSPSS is the first system, other than the original one, to be developed under it.

The UFI portion of the MSPSS, the new subsystem, allows the user-to construct an "experiment" characterized by various algorithms and parameters. Once an experiment has been completely described, a computer program for performing the experiment is automatically generated, compiled, and executed. Since these are simulation experiments, the generated program can run for quite a while. Typically, a single experiment is used to analyze the performance of the signal detection algorithm in terms of the false alarm rate and the probability of detection.

The MSPSS supports the RL investigations described above. Several extensions to this work are described, but not analyzed, in *Multichannel Detection Using the Discrete-Time Model-Based Innovations Approach* [Michels 91]. First, the synthesized processes generated for the signal detection analysis can be generalized or changed. The signal, clutter, and noise, as synthesized by the MSPSS as it existed at the start of this project, were all described by either AR processes or Gaussian white noise. The sum of signal, clutter, and noise is then an Autoregressive Moving Average (ARMA) process. Dr. Michels has suggested generalizing the process synthesis to include nonstationary bandpass processes, processes modeling multipath situations, and spherically invariant random processes (SIRP) for the generation of more general classes of non-Gaussian processes.

Second, the signal detection algorithm can be adapted to handle these new cases. Multichannel likelihood ratios need to be developed and implemented for nonstationary bandpass processes, multipath analysis, adaptive processing, and non-Gaussian SIRP processes.

Finally, the techniques implemented in the MSPSS at the beginning of this project could be enhanced. In particular, additional decomposition methods can be added, as well as new diagnostics for testing the distribution of a process. Dr. Aydin Ozturk, a statistician recently appointed as a visiting professor at Syracuse University has suggested a method for determining probability distributions based on limited data samples. Dr. Ozturk's colleagues at Syracuse University have been examining different methods of applying Ozturk's algorithm. A method of testing distributions based on this work could be added to the MSPSS.

The work reported in this project added functionality to the MSPSS to support the analysis of these generalizations and extensions.

1.2 Objective

The objective of this effort was to enhance the MSPSS to support generalizations of the analysis implemented at the beginning of the project. These generalizations were based on suggestions for future work described in *Multichannel Detection Using the Discrete-Time Model-Based Innovations Approach* [Michels 91]. Specifically, such enhancements were expected to support the investigation of new multichannel signal processing algorithms consisting of the following:

- The synthesis and analysis of multipath and feedback processes as described in Section 8.2.6 of [Michels 91]

- The synthesis and analysis of multichannel extensions of non-Gaussian processes using Spherically Invariant Random Processes (SIRP) as the white noise driving term in an Autoregressive (AR) process.
- Enhancement of the synthesis and analysis procedures to support adaptive processing. In particular, coefficients describing the synthesized process can be abruptly changed.
- Implementation of new algorithms, such as the Singular Value Decomposition of covariance matrices and a goodness-of-fit test.

1.3 Tasks Performed

This effort consisted of tasks that developed new capabilities in process synthesis methods, estimation algorithms, likelihood ratio algorithms, and new diagnostic algorithms. The specific details of these tasks evolved in consultation with RL to provide capabilities that best support their research directions.

1.3.1 Task 1: Process Synthesis

Under this task, four extensions were made to the signal synthesis modules of the Multichannel Signal Processing Simulation Software.

1.3.1.1 Unconstrained Quadrature Components

A synthesis procedure resulting in multichannel autoregressive processes with Gaussian statistics and unconstrained quadrature components was implemented. This procedure requires the use of the correlation function shaping techniques contained in the current software to describe the correlation matrix given in Equation 4.6.3.b of [Michels 91], to solve for the coefficients in the Yule-Walker equation 4.6.6, and to synthesize Autoregressive processes as described in Equation 4.6.21.b.

Unconstrained quadrature synthesis modules were added to both the menu-based old system in the MSPSS and the UFI-based new system.

1.3.1.2 Multipath Processes

The synthesis capability was extended to include the effects of multipath, as described in Figure 8.2.1 of [Michels 91]. In this extension, the signal processes were configured to contain delayed and weighted values of the non-white clutter noise, where the number of delays, the time position, and the values of the weights are user specified.

Multipath synthesis modules were added to both the menu-based old system in the MSPSS and the UFI-based new system.

1.3.1.3 Spherically Invariant Random Processes

Multichannel Non-Gaussian process synthesis algorithms using the Spherically Invariant Random Processes (SIRP) method [Rangaswamy 91] were implemented. This procedure is used to generate a non-Gaussian white noise process as an alternative to the Gaussian white noise driving term in the previous AR process synthesis procedures. The specific approach utilizes the procedure outlined in [Rangaswamy 91] for generating SIRPs for the K distribution.

Detection analyses using SIRP noise and AR clutter being investigated by RL currently use a deterministic model for the signal. A capability of generating deterministic signals was

also added under this task.

SIRP synthesis modules were added to both the menu-based old system in the MSPSS and the UFI-based new system. A miniversion of the 'old system,' focused on SIRP process synthesis, was delivered to Dr. Jorge Romeu, at the direction of RL. This minisystem supported the research described in *Monte Carlo Validation of A Theoretical Model for the Generation of Non-Gaussian Radar Clutter* [Romeu 92].

1.3.1.4 Processes with Abrupt Parameter Changes

An AR process synthesis procedure capable of simulating an abrupt change in the process parameters was implemented. In this procedure, the user has control of the correlation properties of the processes before and after the change, as well as the temporal location of the change.

This capability was implemented by providing a module for concatenating multiple processes under the old system. Each process that is concatenated can be synthesized under a different model, or the same model with different parameters.

The analysis supported by this capability is still a subject of current RL research. The flexibility of the 'old system' is still needed, and canned analysis sequences cannot yet be defined to take advantage of this capability. Therefore, it has not yet been added to the 'new system.'

1.3.2 Task 2: Algorithm Implementation

The Single Value Decomposition (SVD) was implemented as an addition to the previously available Cholesky and LDU decompositions of the error covariance matrix. This capability was implemented in the white (driving) noise generation of process synthesis procedures in the old system. It was also implemented in the linear prediction filters in both the old and new systems.

1.3.3 Task 3: Likelihood Ratio Implementation

Two new likelihood ratio statistics were implemented under this effort. One of these likelihood ratios is appropriate for an innovations-based model in which quadrature components are not constrained to be uncorrelated. The other is appropriate for a deterministic signal in K-distributed SIRP noise.

1.3.3.1 Unconstrained Quadrature Likelihood Ratio

The likelihood ratio detection schemes described in Equations 6.3.27 and 6.3.37 of [Michels 91] were implemented for the case of Gaussian processes with unconstrained quadrature components. This capability was implemented under both the old and new systems.

Also, the constrained quadrature likelihood ratio contained in the previous version were configured under the 'new system' to perform detection analyses using the unconstrained quadrature process synthesis procedures. (With the flexibility in the interaction of modules, this capability exists under the 'old system' with no additional labor.) The constrained quadrature likelihood ratios were derived under the assumption that the in-phase and quadrature components of the radar are uncorrelated. This capability allows performance degradation to be examined as this assumption is increasingly violated.

1.3.3.2 SIRP Likelihood Ratio

A new likelihood ratio and a filtering scheme were implemented for the case of a deterministic signal in SIRP noise or AR clutter with SIRP driving noise. The implementation allows the Gaussian limiting case of the K distribution to be used as well. This capability was implemented in both the new and old systems.

The previously existing loglikelihood ratio derived for Gaussian processes was configured in the new system so that SIRP processes could be evaluated with that loglikelihood statistic as well. Thus, the case of a mis-matched receiver can be addressed. This capability is provided by the structure of the 'old system' with no additional effort.

1.3.4 Task 4: Diagnostics Implementation

A capability was added for estimating the distribution of a quadratic form that characterizes SIRPs. This capability was added to the old system. Since this capability is not appropriate for batch operation and requires a graphical display of the results, it was not implemented under the new system.

1.4 Summary of Results

The capabilities of the MSPSS were considerably generalized by this effort. Previously, the synthesis and analysis capabilities were limited to Gaussian processes with uncorrelated in-phase and quadrature components. New capabilities added under this effort remove this constraint on quadrature components, add an important non-Gaussian capability, support the modeling of multipath processes, and allow process parameters and models to abruptly change. Diagnostic procedures, new loglikelihood ratios, and other new algorithms appropriate for these new models were added to the MSPSS. With these new capabilities, RL/OCTM can explore the robustness, performance, and applicability of the innovations-based loglikelihood ratio in [Michels 91] in much more general circumstances. It will require some time before the full potential of this new system is thoroughly explored.

2. THE MULTICHANNEL SIGNAL PROCESSING SIMULATION SYSTEM

Kaman Sciences has developed the Multichannel Signal Processing Simulation System (MSPSS) to support research being conducted by RL/OCTM. This section briefly overviews the problems that can be examined with the MSPSS and how capabilities are added to this software system.

2.1 The Multichannel Signal Detection Problem

The MSPSS is designed to simulate correlated random vectors which characterize several channels of data. The data gathered at any point in time consists of a (possibly complex) vector with a component for every channel. A single realization of the process generating this data consists of a ordered sequence of such vectors. When no target is present, the data will consist of white noise and, optionally, correlated noise (clutter).

The signal detection problem is to decide if the data is, in fact, white noise and clutter or if a signal is present. That is, one must decide between the null hypothesis

- H_0: $x = c + n$

and the alternative

- H_1: $x = s + c + n$

where x denotes the observed return, s denotes a signal, c denotes clutter, and n denotes noise. The probability of falsely deciding a signal is present is known as the false alarm probability, while the probability of correctly indicating a signal when one is present is known as the probability of detection. (Considered as a problem in the Neyman-Pearson theory of statistical hypothesis testing, the false alarm probability is known as the significance level. The probability of detection is the power of the test.) Typically, signal detection algorithms are evaluated as a function of these two parameters, the false alarm rate and the probability of detection. The MSPSS supports the estimation of the threshold value and probability of detection under user-specified conditions for given false alarm rates.

2.1.1 An Innovations Approach

The MSPSS implements the approach to this problem described in [Michels 89] and [Michels 91]. This is a model-based approach depending on specific models for the signal, clutter, and noise. The signal and clutter are each modeled as Vector Autoregressive (VAR) processes. A VAR process $x(1), x(2), \ldots, x(N)$ is defined by:

$$x(n) = -\sum_{k=1}^{p} A(k)\, x(n-k) + z(n),$$

where $z(n)$ is a random (possibly complex) zero-mean vector with covariance matrix Σ. If the off-diagonal elements of Σ are non-zero, the time samples $x(n)$ are correlated across channels. The AR parameters $A(k)$ lead to correlation of the process in time.

Previously, the distribution of the driving noise term was Gaussian throughout the MSPSS. This effort added a capability to allow the driving noise distribution to be a K-distributed Spherically Invariant Random Process. Also, the processes were previously represented by the in-phase and quadrature components in complex form. That is, if the returns have J channels, $x(n)$ is a J element complex vector:

$$x(n)^T = [x_1(n), x_2(n), \ldots, x_J(n)],$$

where the real component of $x_j(n)$ represents the in-phase component of the jth channel, and the imaginary component represents the quadrature component. This effort added a capability to represent in-phase and quadrature components in concatenated quadrature form:

$$r(n)^T = [x_I{}^T(n), x_Q{}^T(n)],$$

where $x_I(n)$ and $x_Q(n)$ are J-element real vectors representing the in-phase and quadrature components respectively:

$$x_I{}^T(n) = [x_{I,1}(n), x_{I,2}(n), \ldots, x_{I,J}(n)],$$

$$x_Q{}^T(n) = [x_{Q,1}(n), x_{Q,2}(n), \ldots, x_{Q,J}(n)].$$

In addition, a more general procedure was implemented to allow arbitrary correlation between these components.

The remainder of the mathematics in this section does not apply to the concatenated quadrature form of a VAR process.

The covariance matrix of the driving noise, Σ can be decomposed in one of several ways. The Cholesky decomposition is represented by:

$$\Sigma = C\, I\, C^H.$$

C is a lower diagonal matrix, and I is the identity matrix. The LDU decomposition is

$$\Sigma = L\, D\, L^H,$$

where D is a diagonal matrix and L is a lower diagonal matrix with unity along the principal diagonal. At the start of this effort, the MSPSS only supported the Cholesky and LDU decompositions. During this project, the Singular Value Decomposition (SVD) was added:

$$\Sigma = Q\, \Lambda\, Q^H.$$

Λ is a diagonal matrix in which the principal diagonal consists of the eigenvalues of Σ. The columns of Q are the corresponding eigenvectors. The matrix Q is unlikely to be lower diagonal.

Any one of these decompositions can be used to construct an "innovations" representation of a VAR process. Let M be C, L, or Q. The innovations representation is then given by the following equation:

$$M^{-1}x(i) = -\sum_{k=1}^{p} M^{-1}A(k)\,x(i-k) + M^{-1}z(i),$$

where $M^{-1}z(i)$ are the innovations. The random vector $M^{-1}z(i)$ is uncorrelated in time and across channels and has variances given by the diagonal matrix in the corresponding decomposition. At present, the MSPSS supports innovations-representations of autoregressive processes. A state-space capability is currently being developed.

2.1.2 Shaping Functions and the Yule-Walker Equations

A Vector Autoregressive model of the signal or clutter can be specified by the AR parameters and either the covariance matrix of the driving noise or one of the decompositions described above. However, direct specification of these parameters does not readily enable the control of the temporal and cross-channel correlation. Consequently, a "shaping function"

has been proposed for controlling AR parameters in terms of physically meaningful parameters [Michels 91].

For a stationary, zero-mean stochastic process $x(1),x(2),...,x(N)$, one can define a matrix showing the correlations across channels for the kth lag:

$$R_x(k) = E\,[x(n)\,x^H(n-k)],$$

where $x^H(n-k)$ denotes the Hermetian transpose of the vector $x(n-k)$. (The Hermetian transpose of a matrix is the complex conjugate, on an element by element basis, of the transpose of a matrix.) Note that if the number of channels is J, $R_x(k)$ is a J x J matrix. Furthermore, it follows from this definition that the following holds:

$$R_x(-k) = R_x{}^H(k)$$

The correlation matrix and the AR parameters are related by the Yule-Walker equations. For example, if the number of lags, p, is three, the Yule-Walker equations are as follows:

$$\begin{bmatrix} I & A(1) & A(2) & A(3) \end{bmatrix} \begin{bmatrix} R(0) & R(1) & R(2) & R(3) \\ R(-1) & R(0) & R(1) & R(2) \\ R(-2) & R(-1) & R(0) & R(1) \\ R(-3) & R(-2) & R(-1) & R(0) \end{bmatrix} = \begin{bmatrix} \Sigma & 0 & 0 & 0 \end{bmatrix}$$

Given the matrix of correlations R, these equations can be solved for the AR parameters and the driving noise covariance matrix Σ.

The user of the MSPSS does not specify the correlation matrix for each lag, but certain parameters of "shaping" functions" from which the correlation matrix is calculated. Two forms of shaping functions are available, a Gaussian and an Exponential function. The Gaussian shaping function is defined by

$$R_{x,i,j}(k) = K_{i,j}\,\frac{\lambda_{i,j}^{[(k-l_{i,j})^2]}}{\lambda_{i,j}^{(l_{i,j}{}^2)}}\,e^{2\pi\phi T k\sqrt{-1}},\ i \le j$$

$$R_{x,i,j}(k) = R_{x,j,i}{}^*(-k),\ i > j.$$

K is the amplitude matrix, λ is the one-lag temporal correlation parameter, l is the lag value at which the function peaks, ϕ is the Doppler shift, and T is the sampling period. X^* is the complex conjugate of the complex number X.

The Exponential shaping function is defined by

$$R_{x,i,j}(k) = K_{i,j}\,\frac{\lambda_{i,j}^{|k-l_{i,j}|}}{\lambda_{i,j}^{|l_{i,j}|}}\,e^{2\pi\phi T k\sqrt{-1}},\ i \le j$$

$$R_{x,i,j}(k) = R_{x,j,i}{}^*(-k),\ i > j.$$

2.1.3 The Signal Detection Algorithm

An innovations-based signal detection algorithm has been defined for the above model [Michels 91]. The signal and clutter are assumed to be VAR processes, while noise is white and possibly correlated across channels. The sums of the signal, clutter, and noise are Autoregressive Moving Average (ARMA) processes. These can be approximated by different high order VAR processes.

Given the parameters of the VAR process, including the decomposition of the driving noise covariance matrix, a linear filter can be constructed whose outputs are the innovations when the inputs are the VAR process. Two such filters, denoted F0 and F1, need to be constructed for this signal detection algorithm. The parameters of the VAR process approximating the ARMA process for the null hypothesis describe the F0 filter. Thus, when the input is the sum of clutter and noise, the output of the F0 filter will be white noise uncorrelated in time and across channels. Similarly, the F1 filter applies to the alternative hypothesis. When the input is the sum of signal, clutter, and noise, the output of the F1 filter will be uncorrelated white noise. The outputs of the two filters, then are estimates of the innovations for the VAR models corresponding to the null and alternate hypotheses.

The signal detection algorithm works by first filtering two copies of the returns by the two filters F0 and F1 in parallel. A loglikelihood ratio is then calculated on the outputs of these filters. If the loglikelihood statistic is above a predefined threshold, the algorithm allows one to decide a signal is present. Otherwise, the alternative hypothesis should be rejected and one should decide no signal is present.

This algorithm raises many questions that the research program being conducted by RL/OCTM is designed to answer. What threshold values should be used for given false alarm rates and given parameters of the signal, clutter, and noise models? What is the probability of detection? How does the performance vary with different algorithms for estimating the filter coefficients and different estimation algorithms (e.g. the Yule-Walker, the Nuttall-Strand, and the Vieira-Morf algorithms)? Does the use of different decomposition algorithms (Cholesky, LDU, or SVD) alter the performance? How does the algorithm perform for different varieties of non-Gaussian noise? How much variation is there in the performance of these algorithms as the process correlation changes? When are AR models good approximations? What happens if the models embodied in the filters do not exactly match the radar returns?

The MSPSS provides the analyst with tools to answer these questions via Monte-Carlo simulation. Capabilities are provided for synthesizing signal, clutter, noise, and their sum. In process synthesis, the user enters the parameters of the shaping functions and chooses a decomposition method. The program solves the Yule-Walker equations by means of the Levinson-Wiggins-Robinson algorithm. The program synthesizes the innovations for an AR process, multiplies the innovations by the decomposed covariance matrix, and adds the weighted sum of previous lags. A user-specified number of realizations of signal, clutter, and noise is generated, where the number of time samples is also user specified.

Capabilities are provided for performing certain diagnostic functions on the resulting realizations of the generated stochastic processes. The user can calculate means and variances, fast fourier transformations, correlation functions, or estimate autoregressive parameters. The functions, and the synthesized radar returns, can be displayed graphically under an X windows interface. Various functions are provided for manipulating radar returns such as adding them, splitting a multichannel return into several single channel returns, combining single channels into multichannel returns, and so on.

Finally, the MSPSS implements the signal detection algorithm described above. Linear prediction filters are provided, with user-definable parameters, for removing intertemporal and interchannel correlation. Several loglikelihood statistics can now be calculated, depending on the specific model the user chooses. Both thresholds and detection probabilities can be calculated, as well as the variance in these estimates.

2.2 Using the MSPSS

When invoking the original menu-based system, the user is presented with a main menu with two sets of three submenus. The two sets of menus correspond to capabilities for single-channel and multichannel models. The three submenus for each set provide synthesis, filtering, and diagnostic capabilities, respectively. The user chooses a menu option by typing an identifier, followed by a carriage return. Each option in a subprogram corresponds to an individual program. These programs prompt the user for their inputs, echo their outputs to the user's terminal, and save some of their inputs to user-specified binary files. These binary files are readable by other programs in the system.

The new UFI-based system interacts with the user by means of a windowing, icon, and mouse-driven interface. This interface is implemented under X Windows, but a menu-based interface is also provided. The user defines a "project" as a collection of "experiments." An experiment consists of an analysis sequence. A typical analysis sequence will estimate AR parameters for filters, repetitively synthesize noise so as to estimate threshold values, and synthesize signal plus noise so as to estimate detection probabilities. When defining the sequence, the user defines all required inputs. The system is designed such that no inputs require any knowledge of intermediate results.

After defining an experiment, the user can exit the system. The back end then creates a Fortran 77 program implementing this experiment. The program is created, and its execution is begun in background. The system sends a message to the user when finished, using the Unix system's mail facility. The user can then examine the output of the program, which is saved to a file. The use of the new system is described in much greater detail in [Kaman 92a].

2.3 Adding Capabilities to the Old System

The original menu-based system consists of a collection of programs called by c-shells that implement menus providing a simple user-interface. Hence, new capabilities are added by first writing or modifying a main program. These programs are written in the Unix Fortran Preprocessor (UFP) language, a Kaman-internal Ratfor-like language [Robbins 89].

When writing such programs, the analyst should follow the style of existing programs. In particular, an extensive library of subroutines is already available. Some of these subroutines support writing and reading returns to binary files in a standard manner such that all programs in the system can interact with these files. Autoregressive process parameters and channel variances can also be written to binary files.

Fortran format statements and fortran input-output should not appear in any main program. Subroutines are available for echoing messages to Unix "standard output" and "standard error." Parameters should be read from the user by use of these subroutines and external "help" files. In short, new capabilities should be added by leveraging the existing body of tools, which greatly simplify some of the most tedious aspects of programming.

Once a new program and its corresponding help file are written, it must be linked into both the developmental and the delivered version of the system. Obviously, it should first be added to the developmental version and tested before delivery. Programs should be stored in files whose names end with a ".u" extension; for example, "madd.u" is the name of a program for adding two or more multichannel returns together. The corresponding help file should be named with a ".hlp" extension, for example, "madd.hlp".

The developmental version of a program should be stored in the directory ~mmi/mchan on Kaman Science's system. The help files should be in the directory ~mmi/mchan/hlp. The delivered version should be stored in the directory ~kaman/mchan on the computer system operated by RL/OC, and the help files are in ~kaman/mchan/hlp.

Each new program should be added to the "make files" in ~mmi/mchan and ~kaman/mchan. When a program is preprocessed, compiled, and linked, various files will be created. Those files with ".f" extensions contain fortran source code corresponding to the UFP programs in ".u" files. File with ".o" extensions contain object code, while files with no extension contain executable images. Finally, some directories contain files with ".a" extensions. These files are archived libraries of object code.

The last step in adding a program to the original system is adding it to the menus that provide the user-interface. These menus are implemented as c-shells. They are in the directories ~mmi/mchan/bin and ~kaman/mchan/bin.

2.4 Adding Capabilities to the New System

The new system is implemented under the User Front-end Interface (UFI), a tool developed by Kaman to encourage software reusability. A system built under the UFI consists of a collection of reusable UFP subroutines. Ideally, the user of an UFI-based system, who may have no programming experience, describes an experiment. The experiment combines capabilities from the library of reusable components. The order in which functions are called is user-definable. The back-end to a UFI-based system automatically generates and executes a main program implementing the experiment by calling upon the reusable subroutine library. The generated main program is also in the UFP language.

The MSPSS is too complicated to fully support this idealized flexibility. The UFI-based system provides a set of predefined analysis sequences for generating programs for experiments. This system provides the user with a batch capability, with the attendant advantages of allowing the user to quickly and efficiently define an analysis. Such an analysis or experiment corresponds to the invocation of menu items in the first system in a fixed order with user-defined values for all parameters and options. The output of an experiment typically consists of Monte Carlo estimates of the threshold values of the loglikelihood statistic and the probabilities of detection, given false alarm rates and a description of the signal, clutter, and noise.

Capabilities are added to the new system by adding reusable subroutines to the library and by modifying or creating new analysis sequences. Reusable subroutine modules must follow certain interface conventions. The UFI Knowledge Base (UFIKB) specifies their interface, both with the main program and with respect to user-defined parameters. Analysis sequences, also specified in the UFIKB, defines the order in which modules are invoked. Thus, the steps needed to implement a sequence under the MSPSS are:

- Create all source modules.
- Describe all source modules to the UFIKB.
- Write a sequence for the UFIKB.
- Add the sequence name to an appropriate menu.

Existing modules and sequences provided models used by programmers during this effort.

2.4.1 Source Modules

Each main program in the original system is usually decomposed into several modules in the UFI-based system. This decomposition separates those portions of the original program that read parameters from the user and otherwise initialize various entities from those portions that process each realization of a radar return. By convention, modules in the new system are named xxxP, xxxL, and xxxT, where xxx is the main program's name. P stands for "pre-loop," L stands for "loop," and T stand for "Terminate". They are not all always necessary and, in particular, the "terminate" routine often does not exist. Only subroutines that will be called from UFI-generated main programs need follow these conventions.

Typically, the pre-loop program reads all needed variables from the user and passes these variables and related ones (such as the inverse of a matrix) to the loop program. In practice, several pre-loop programs should sometimes be written, supporting several combinations of user-inputs and initialization functions. This supports greater flexibility in nesting loops and more complicated analysis sequences than were originally envisioned for the MSPSS.

The loop program performs some processing that is in a loop in a program which processes every realization or trial of a radar return. Examples of functions in loop programs include process synthesis, estimation, and filtering. In effect, the pre-loop and terminate routines are written merely to support the major functions provided by the loop programs. Loop programs should avoid sending messages to standard output and standard error.

The terminate routine does whatever post processing needs to be done, such as printing out averages of parameter estimates.

The interfaces to these subroutines follow certain conventions. The general pattern is

subroutine XXXY(HDR1, STATE, HDR2, ARRY1, HDRn, ARRYn, IER)

where the headers contain information on the size of the following argument; STATE is a vector of "state" variables (all of the same type); the arrays are such things as a multichannel signal, a vector of variances for each channel, or an AR coefficient (a square matrix in the multichannel case); and IER is an error flag that returns a nonzero value if a problem exists. Macros exist for manipulating the headers in UFP programs.

Existing modules provide models illustrating these conventions in greater detail. The UFP source code for these modules is stored in files with ".u" extensions in the directory ~mmi/src, for the developmental version of the MSPSS. The delivered source is in the directory ~kaman/mmi/src on the RL/OC computer system. When new files are added to these directories, they should be described to the make files in these directories as well.

2.4.2 Knowledge Base

Once a UFI module is written, it is described to the knowledge base. The knowledge base description specifies the arguments for each subroutine and what parameters need to be read from the user. The UFI uses this specification to prompt the user for these parameters when the user is defining an experiment. A file is created by the UFI for storing these parameters. It is this file that is actually read by generated programs when they are executing.

The *UFIKB Builders' Guide* [Kaman 91b] describes the knowledge base language in detail, while existing knowledge base files can be used as models. Each UFP source program

xxx.u should have a corresponding knowledge base file named xxx.mkb. These knowledge base files are in the directory ~mmi/kb or ~kaman/mmi/kb. Acceptable types for arguments in subroutine interfaces are defined in the file "ioblock.mkb."

Once a new knowledge base file is written, it must be entered into the make file in the directory in which it is stored. It must also be entered into the file "mmi.mkb," which lists all those files comprising the local knowledge base of the UFI-based version of the MSPSS.

2.4.3 Sequences and Menus

Sequences are also written in the UFIKB language. Sequences are added to the UIFKB in a manner similar to that used for modules. They are defined in files with an extension of ".mkb". These files must be added to the make file and to the file "mmi.mkb." In addition, each sequence should appear on a menu. These menus are defined in the file "xallmenu.ikb."

3. NEW CAPABILITIES

Under this effort, new capabilities were added to the Multichannel Signal Processing Simulation System (MSPSS) in the areas of

- Process synthesis methods
- Estimation algorithms
- Likelihood ratio algorithms
- Diagnostic algorithms

New capabilities were first implemented in the original menu-based system following the procedures described in Section 2.3. When appropriate, they were then implemented in the new UFI-based system following the procedures described in Section 2.4. This section further defines the capabilities added during this effort. The capabilities implemented under the original menu-based system are first described in detail. The analysis sequences implemented under the new UFI-based system are briefly outlined in the next section.

3.1 Process Synthesis

Four additional programs were designed under this effort for process synthesis. Two relate to unconstrained quadrature synthesis. The third is a very general synthesis program that includes options for Gaussian processes, Spherically Invariant Random Processes (SIRPs), and multipath processes. The following sections describe these programs in detail. These sections are modifications of the memos that were written specifying the programs before they were developed.

3.1.1 Quadrature Synthesis

The two programs for synthesizing processes with unconstrained quadrature components are each callable from a menu choice in the original system. The first synthesizes the sum of signal, clutter, and noise, where the signal and clutter are specified by a shaping function approach. The output is in concatenated quadrature form and is stored in files that can be read by the other programs in the system. The other program converts radar returns in concatenated quadrature form to complex form.

The synthesis program generates a multichannel radar return in concatenated quadrature form as discussed in Section 2.1.1. The radar return consists of a signal, clutter, and noise. The user has the option of not including signal, clutter, and noise. The signal and clutter, if present, consist of Autoregressive (AR) processes with Gaussian driving noise. The noise is Gaussian white noise.

Specifically, the program generates the sequence of real column vectors $r(1)$, $r(2)$, ..., $r(N)$, where

$$r(n) = s(n) + c(n) + w(n), n = 1, 2, ..., N. \quad (1)$$

The sequence of vectors $r(n)$ represents the return process. The sequence of vectors $s(n)$ represents the signal. $c(n)$ represents the clutter, and $w(n)$ represents the noise.

The number of elements in each vector $r(n)$, $s(n)$, $c(n)$, $w(n)$ is equal to twice the number of channels, J. Each vector $r(n)$ can be thought of as a 2 element column vector where each element is itself a column vector:

$$r(n) = \begin{bmatrix} x_I(n) \\ x_Q(n) \end{bmatrix}.$$

$x_I(n)$ are the in-phase components, and $x_Q(n)$ are the quadrature components of the process. They are real column vectors. For example, if the process contains two channels, the in-phase and quadrature components are each composed of two real numbers:

$$x_I(n) = \begin{bmatrix} x_{I,1}(n) \\ x_{I,2}(n) \end{bmatrix}, \quad x_Q(n) = \begin{bmatrix} x_{Q,1}(n) \\ x_{Q,2}(n) \end{bmatrix}.$$

This representation of a radar return is known as the concatenated quadrature form.

A single sequence of vectors $r(1)$, $r(2)$, ..., $r(N)$ is referred to as a *trial* or *realization* of the process. Since random processes are used in generating radar returns, the radar returns will vary from trial to trial. Each vector, $r(n)$, in the sequence of vectors is referred to as a *time sample*. Once again, because of the random nature of the process, the N time samples will vary across a single realization of the underlying process.

3.1.1.1 White Noise

The Gaussian noise vector, $w(n)$, is characterized as a zero-mean process for both the in-phase and quadrature components of each channel and a square complex covariance matrix Σ_w. The noise is not correlated across time, but may be correlated across channels. The covariance matrix Σ_w specifies the cross-channel correlations. It is defined as the following:

$$\Sigma_w = E[w(n)\, w^H(n)].$$

One can think of the covariance matrix as a matrix of matrices:

$$\Sigma_w = \begin{bmatrix} \Sigma_{w,II} & \Sigma_{w,IQ} \\ \Sigma_{w,QI} & \Sigma_{w,QQ} \end{bmatrix}.$$

In all other synthesis programs in this system, *the QI and IQ components of covariance and correlation matrices are zero. This program differs from these other synthesis programs in that it provides the user with explicit control of these correlations.*

The user does not specify the covariance matrix. Rather, either the Cholesky, the LDU, or the Singular Value Decomposition (SVD) of the covariance matrix is specified. The Cholesky decomposition is defined to be

$$\Sigma = (1/2) C\, C^H,$$

where the factor of $1/2$ is included to divide the variance of each channel among the In-phase and Quadrature components. C is a lower triangular matrix, and C^H is its Hermetian transpose. The LDU decomposition is

$$\Sigma = (1/2) L\, D\, L^H.$$

L is a lower triangular matrix with ones along the principal diagonal, and D is a diagonal matrix used to control the variances. The SVD is

$$\Sigma = (1/2) Q\, \Lambda\, Q^H.$$

Λ is a diagonal matrix. Each element of Λ is an eigenvalue of 2Σ, while the columns of Q are the corresponding right hand eigenvectors. The rows of Q^H are left hand eigenvectors. The eigenvectors in Q should have a norm of unity. Also, Q^H is the inverse of Q, that is,

$$Q^H = Q^{-1}$$

The user specifies either C, L and D, or Q and Λ. For each time sample in each trial, the program synthesizes a zero mean vector:

$$t(n) = \begin{bmatrix} t_I(n) \\ t_Q(n) \end{bmatrix}.$$

The elements of the in-phase and quadrature vectors comprising $t(n)$ are statistically independent. If the user has specified a Cholesky decomposition, each element has a variance of 1/2. Otherwise the variance of the jth element of $t(n)$, $j = 1, 2, \ldots, 2J$, is $(1/2)D_{j,j}$ or $(1/2)\Lambda_{j,j}$.

The noise is then generated by the appropriate formula, depending on the user-specified decomposition:

$$w(n) = C\,t(n).$$

$$w(n) = L\,t(n).$$

$$w(n) = Q\,t(n).$$

When decompositions are echoed to the user, a message is also written reminding the user of the factor of 1/2. (A factor of 1/2 does not appear in decompositions involving the complex process synthesis routines.)

3.1.1.2 AR Process Generation

The signal and clutter are generated as AR processes:

$$x(n) = -\sum_{k=1}^{P} A(k)x(n-k) + v(n),$$

where $A(1)$, $A(2)$, ..., $A(P)$ are the AR coefficients, P is the order of the AR process, and $v(n)$ is the driving white noise term with covariance matrix Σ_v. The driving noise is generated much as the noise is generated above based on the decomposition the user chooses.

Each AR coefficient can be thought of as a matrix of matrices:

$$\begin{bmatrix} x_I(n) \\ x_Q(n) \end{bmatrix} = -\sum_{k=1}^{P} \begin{bmatrix} A_{II}(k) & A_{IQ}(k) \\ A_{QI}(k) & A_{QQ}(k) \end{bmatrix} \begin{bmatrix} x_I(n-k) \\ x_Q(n-k) \end{bmatrix} + \begin{bmatrix} v_I(n) \\ v_Q(n) \end{bmatrix}$$

The user does not specify the AR coefficients or the covariance matrix. Rather, they are found by solving the Yule-Walker equations. As an example, if P is three, the Yule-Walker equations look like the following:

$$\begin{bmatrix} I & A(1) & A(2) & A(3) \end{bmatrix} \begin{bmatrix} R(0) & R(1) & R(2) & R(3) \\ R(-1) & R(0) & R(1) & R(2) \\ R(-2) & R(-1) & R(0) & R(1) \\ R(-3) & R(-2) & R(-1) & R(0) \end{bmatrix} = \begin{bmatrix} \Sigma & 0 & 0 & 0 \end{bmatrix}$$

The correlations matrices $R(k)$ are themselves matrices of matrices. The submatrices correspond to II, IQ, QI, and QQ correlations. The user controls these matrices by specifying the parameters of certain "shaping functions" used to calculate the matrix values.

Two shaping functions are provided, Gaussian and Exponential. The Gaussian shaping function is of the following form:

$$R^{AB}{}_{x,i,j}(l) = K^{AB}{}_{i,j} \frac{\lambda^{AB}{}_{i,j}{}^{[(l-l^{AB}{}_{i,j})^2]}}{\lambda^{AB}{}_{i,j}{}^{(l^{AB}{}_{i,j}{}^2)}}.$$

The Exponential function is of the following form:

$$R^{AB}{}_{x,i,j}(l) = K^{AB}{}_{i,j} \frac{\lambda^{AB}{}_{i,j}{}^{|l-l^{AB}{}_{i,j}|}}{\lambda^{AB}{}_{i,j}{}^{|l^{AB}{}_{i,j}|}}.$$

The superscripts A and B can be either I or Q. The subscripts i and j range over the number of channels. K^{AB} is the amplitude, λ^{AB} is the one-lag temporal correlation parameter, and l^{AB} is the lag value at which the function peaks. The user enters these values.

Unlike the shaping functions used in the procedures for synthesizing returns in complex form, these shaping functions do not include parameters for Doppler shifts.

Subroutines for performing SVDs were added to the system during this effort. They were obtained from LINPACK [Dongarra 79].

3.1.1.3 Implementation Notes

The program asks the user for parameters describing the signal, clutter, and noise. It calculates the AR parameters from the solution of the Yule-Walker equations, and decomposes the covariance matrix, as appropriate. A user-specified number of realizations of the signal, clutter, and noise are generated and summed, as specified. The resulting trials are stored in a user-specified file.

To ensure compatibility with other programs in the system, the file containing the radar return written by this program stores each trial in a complex array with N rows and $2J$ columns. N is the number of time samples. Hence each row corresponds to a single time sample. J is the number of channels in the process. The first half of each row represents the in-phase components of a time sample, and the second half represents the quadrature components. The imaginary part of all elements of this array are identically zero. We note that the input file is readable by the subroutine MCIMC, the same subroutine used to read radar returns in other programs in this system.

Likewise, AR coefficients and covariance matrices are written in a form compatible with other programs in the system.

3.1.2 Conversion to Complex Form

A program was written to convert the concatenated quadrature representation of a radar return to the complex representation. The routine asks the user for the names of an input file and an output file. The routine reads a number of trials from the input file, where each trial consists of a number of time samples from a multichannel radar return in concatenated quadrature form. It converts each trial to the complex representation, and writes the result to the output file in a form readable by other programs in this system.

Let $x(1)$, $x(2)$, ..., $x(N)$ be a single trial read from the input file. Then $x(n)$ is a 2 element column vector where each element is itself a column vector:

$$x(n) = \begin{bmatrix} x_I(n) \\ x_Q(n) \end{bmatrix}$$

$x_I(n)$ are the in-phase components, and $x_Q(n)$ are the quadrature components of the process. They are real column vectors. For example, if the process contains two channels, the in-phase and quadrature components are each composed of two real numbers:

$$x_I(n) = \begin{bmatrix} x_{I,1}(n) \\ x_{I,2}(n) \end{bmatrix}, \quad x_Q(n) = \begin{bmatrix} x_{Q,1}(n) \\ x_{Q,2}(n) \end{bmatrix}.$$

To ensure compatibility with other programs in the system, the file containing the input signal stores each trial in a complex array with N rows and J columns. In other words, the quadrature form is stored in the same data structure as the complex form. This leads to a reinterpretation of the elements of the complex form. N is the number of time samples. Hence each row corresponds to a single time sample. J should be an even number, and is twice the number of channels in the process. The first half of each row represents the in-phase components of a time sample, and the second half represents the quadrature components. The imaginary part of all elements of this array are identically zero. Note that the input file is readable by the subroutine MCIMC, the same subroutine used to read radar returns in other programs in this system.

The complex form of a two channel vector is expressed as:

$$x(n) = \begin{bmatrix} x_{I1}(n) + j\,x_{Q1}(n) \\ x_{I2}(n) + j\,x_{Q2}(n) \end{bmatrix},$$

where j is the square root of negative one. The subroutine MCOMC provides a suitable means of writing trials to the output file in a format compatible with the rest of the system.

3.1.3 SIRP Synthesis

A program was written to provide a wide range of synthesis capabilities associated with Spherically Invariant Random Processes (SIRPs) and multipath processes. This routine generates a multichannel radar return consisting of a signal, clutter, and white noise, with the user having the option of not including any of these. The signal, if present, consists of either

- A deterministic signal
- An Autoregressive (AR) process with either a multichannel Spherically Invariant Random Process (SIRP) or Gaussian noise driving term, or
- An AR-like multipath process

The clutter, if present, consists of an AR process with either SIRP or Gaussian driving noise. The white noise is either SIRP or Gaussian.

Specifically, the routine generates the sequence of vectors $r(1), r(2), \ldots, r(N)$, where

$$r(n) = s(n) + c(n) + w(n), \; n = 1, 2, \ldots, N.$$

The sequence of vectors $r(n)$ represents the return from the radar. The sequence of vectors $s(n)$ represents the signal. $c(n)$ represents the clutter, and $w(n)$ represents the white noise. The number of elements in each vector $r(n)$, $s(n)$, $c(n)$, $w(n)$ is equal to the number of channels, J.

A single sequence of vectors $r(1), r(2), \ldots, r(N)$ is referred to as a *trial* or *realization* of the process. Since random processes are used in generating radar returns, they will vary

from trial to trial. Each vector, $r(n)$, in the sequence of vectors is referred to as a *time sample*. Once again, because of the random nature of the process, the N time samples vary across a single realization of the underlying process.

3.1.3.1 Deterministic Signal

Deterministic signals are represented by

$$s_i(n) = A_i \cos[\theta_i(n) + \phi_i] + j B_i \sin[\theta_i(n) + \phi_i],$$

where A_i and B_i are user-specified complex constants; $\theta_i(n) = 2\pi(n-1)(f_d T)_i$, and $(f_d T)_i$ is a user-specified constant between -0.5 and +0.5; j is the square root of -1; and ϕ_i is the initial phase. ϕ_i can either be a user-specified constant between 0 and 2π, or a random number between 0 and 2π.

Strictly speaking, if the phase is random, the signal is not deterministic. Since all other parameters are deterministic, and this option is a special case, the signal is described as a "generalized deterministic signal." The constants A_i and B_i are complex instead of real to give the user more flexibility. In particular, if the user sets all parameters except A_i to zero, the user can easily specify a complex constant value for the signal.

3.1.3.2 Noise

Noise can either be Gaussian white noise or a SIRP.

3.1.3.2.1 Gaussian Noise

The Gaussian noise vector, $w(n)$, is characterized as a zero-mean process for each channel and a square complex covariance matrix Σ. Physically, the diagonal elements of Σ are the variances of the channels, and the off-diagonal elements are the cross-channel correlation coefficients. For example, $\Sigma_{i,j}$ is the variance of channel i. The number of rows (and columns) in Σ is equal to the number of channels.

The user specifies either the Cholesky decomposition, an LDU decomposition, or a Singular Value Decomposition (SVD) of the covariance matrix. The Cholesky decomposition is

$$\Sigma = C\,C^H,$$

where C is a lower triangular complex matrix, and C^H is its Hermetian transpose. The LDU decomposition is

$$\Sigma = L\,D\,L^H$$

where L is a lower triangular matrix with ones along its diagonal and D is a diagonal matrix. The SVD is

$$\Sigma = Q\,\Lambda\,Q^H.$$

Λ is a diagonal matrix. Each element of Λ is an eigenvalue of 2Σ, while the columns of Q are the corresponding right hand eigenvectors. The rows of Q^H are left hand eigenvectors. The eigenvectors in Q should have a norm of unity. Also, Q^H is the inverse of Q. The user specifies either C, L and D, or Q and Λ.

The vector $w(n)$ is generated as follows. First, for each time sample in a realization, generate a vector, $z(n)$, of independent zero-mean normally distributed random variates. The ith channel has a variance of either unity, if the user specified a Cholesky decomposition; $D_{i,i}$,

if the user specified an LDU decomposition; or $\Lambda_{1/2}$, if the user specified a SVD. Then the white noise vector $w(n)$ is found by the appropriate equation:

$$w(n) = C \cdot z(n)$$

$$w(n) = L z(n).$$

$$w(n) = Q z(n).$$

3.1.3.2.2 SIRP Noise

The SIRPs in this routine are specified by one of the above decompositions of the covariance matrix Σ, as above, and two additional parameters, α and b. For SIRPs, let $z(n)$ be a vector of independent zero-mean normally distributed random variates. The SIRP noise is then

$$w(n) = C S z(n),$$

$$w(n) = L S z(n),$$

or

$$w(n) = Q S z(n),$$

where S is a random diagonal matrix generated as explained below.

The user has two choices to make when specifying the generation of S. First, S can be constant for a single realization (but vary across realizations), or S can vary from time sample to time sample in each realization. Second, the diagonal elements of S can all be equal, or a different variate can be generated for each nonzero element of S.

To generate an element of S the program follows the algorithm given in [Rangaswamy 91]. First, it generates a Chi Squared random variate u with 2α degrees of freedom. u is generated based on an algorithm for Gamma variates in [Fishman 73]. It tends to create overflow and underflow problems for $\alpha < 0.25$ or $\alpha > 100.0$. Warning messages alert the user. Also the algorithm tends to become much slower as the number of degrees of freedom increases.

Once the program has generated a Chi Squared variate, it then calculates a random variate v:

$$v = \frac{\sqrt{u}}{b}.$$

v is a random variable with the following Probability Density Function (PDF):

$$f_v(v) = \frac{2b}{\Gamma(\alpha)2^{\alpha}} (b\,v)^{2\alpha-1} e^{-b^2 v^2/2}.$$

Let the random variable s be defined as follows:

$$s = \frac{v}{a},$$

where

$$a = \sqrt{2\alpha/b^2}.$$

Then s is the desired element for the random matrix S.

s has unit variance. Thus, the matrix S has the identity matrix as its covariance matrix. Therefore, the covariance matrix for SIRP noise remains Σ.

Notice that α and b are constants that do not vary with channels, unlike most other parameters for these multichannel processes.

3.1.3.3 AR Process

In the previous section, we discussed how a deterministic signal and white noise are generated. The signal can also be (and the clutter is) an AR process with either SIRP or AR driving noise. An AR process is generated following the "shaping function" approach explained in Section 2.1. That is, The vector stochastic process $x(1), x(2), \ldots, x(N)$ is an Autoregressive process of order m if

$$x(n) = -\sum_{k=1}^{m} A_k x(n-k) + v(n),$$

where $v(n)$ is either SIRP noise or Gaussian noise, as above, with covariance matrix Σ. Note the AR coefficients, $A_1, A_2, \ldots, A_k$, are square matrices. Therefore, given the covariance matrix of the driving noise, the parameters α and b of the SIRP if necessary, and the AR coefficients, the driving noise and the AR process can be generated as above.

In this approach, the following parameters are specified and used in calculating correlations across channels and across time:

- The variance of each channel process,
- The one-lag temporal correlation parameters,
- The lag values at which the cross-channel correlation functions peak,
- The Doppler frequency,
- The sampling frequency.

The AR coefficients and the covariance matrix of the driving noise are found from the correlation matrix by solving the Yule-Walker equations, using the Levinson-Wiggins-Robinson algorithm.

3.1.3.4 Multipath Processes

The signal, if clutter is present, can also follow a multipath model. Previously, the signal was generated as

$$s(n) = -\sum_{k=1}^{m} A_k s(n-k) + v_s(n)$$

Now the user is able to specify matrices B_k and generate the signal as

$$s(n) = -\sum_{k=1}^{m} A_k s(n-k) + \sum_{k=1}^{K} B_k c(n-\wedge) + v_s(n)$$

The matrices A_k and the covariance matrix of the driving noise are still found by the shaping function approach. The user must explicitly enter values for the matrices B_k. These matrices account for a correlationof the signal with past values of clutter

3.1.3.5 Program Input

This routine requires the following input:

- The random number seeds (optional).
- The name of one or more output files.
- Parameters that apply to the radar return in general: the number of trials, the number of time samples per trial, the number of channels, and the number of points to generate and eliminate to avoid the transient period in the AR process synthesis.
- A response as to whether the user wants a signal, and if so, whether the signal should be deterministic, an AR process, or a multipath process. If the user wants an AR or multipath process, whether the driving noise should be a SIRP or Gaussian. In addition, the user must enter appropriate parameters to define the signal.
- A response as to whether the user wants clutter. If so, the user must enter appropriate parameters for its generation.
- A response as to whether the user wants noise. If so, the user must enter appropriate parameters for its generation.
- The names of the files to which the trials of the signal, the clutter, the noise, their sum, and the AR coefficients are written.

The program saves the specified outputs to the specified files.

3.1.4 The ASCII File Minisystem

During this effort, a scaled-down version of the original menu-based system was delivered to Dr. Jorge Romeu to support the research described in [Romeu 92]. This minisystem contained Fortran source code for synthesizing SIRPs and Gaussian processes and for converting binary files containing the synthesized processes to a format suitable for Dr. Romeu's software.

The conversion program reads a file containing a synthesized process. Typically, this file is produced by the single channel SIRP synthesis program or the program for calculating a quadratic form which serves as a sufficient statistic for characterizing SIRPs. It produces an ASCII file.

The program asks the user for a file name. It reads the file and produces a properly formatted output file. Each trial is written on a single line, so the output file has as many lines as trials. The imaginary part of all numbers is discarded. Each line consists of a series of real numbers separated by spaces. The real numbers are not written in exponential format, but rather with a reasonable fixed number of digits to the left and right of the decimal point.

The programs contained in the single channel minisystem delivered to Dr. Romeu are described in Appendix A.

3.1.5 Abrupt Parameter Changes

To give the user an ability to synthesize processes with abrupt parameter changes, a program was written to concatenate up to four files of synthesized data together into a single file. For example, let $x(1), x(2), \ldots, x(N_1)$ and $y(1), y(2), \ldots, y(N_2)$ be two returns. Then the program will create a process $z(1), z(2), \ldots, z(N)$, where

$$N = N_1 + N_2$$

and

$$z(n) = \begin{cases} x(n), n = 1,2,\ldots,N_1 \\ y(n - N_1), n = N_1 + 1, N_1 + 2,\ldots,N \end{cases}$$

If the input files do not contain the same number of trials, the output file will contain the minimum of the number of trials in each input file. The function of the program when the number of channels varies among input files depends on the user. If the user has turned the "zero-fill" option off, the output file will contain returns with the number of channels set to the minimum over the number of channels in all input files. If the user has chosen to zero fill, the number of channels is set to the maximum over the number of channels in all input files. Input files with fewer channels are padded with additional channels containing all zeros.

3.2 Algorithms

As was discussed above in Section 2.1.1, a new matrix decomposition algorithm was added to the MSPSS under this effort. Previously, only the Cholesky and LDU decompositions were supported for diagonalizing the covariance matrix. The Singular Value Decomposition (SVD) is now included in the MSPSS.

The SVD is based on the theory of eigenvalues and eigenvectors. Let M be a matrix. Then λ is an eigenvalue and the column vector x is the corresponding eigenvector if and only if

$$Mx = \lambda x .$$

An n x n matrix always has n eigenvalues, some of which may repeat.

The SVD of a matrix M is given by

$$M = Q \Lambda Q^{-1},$$

where Λ is a diagonal matrix whose principal diagonal consists of the eigenvalues of M. The columns of Q are the corresponding eigenvectors.

The SVD was implemented based on a collection of subroutines in LINPACK [Dongarra 79]. The SVD is a user-selectable option in the synthesis routines, as noted in Section 3.1, and in the linear filter. Only the SVD capability in the linear filter was transferred into the new UFI-based system.

3.3 Likelihood Ratios

Two new loglikelihood ratios were implemented under this effort. One is designed for detecting an AR signal in AR clutter and white noise, where signal, clutter, and noise are modeled in concatenated quadrature form. The other is suitable for detecting a constant signal in AR clutter and white noise, where the driving noise term in the AR clutter and the white noise are both modeled by SIRPs.

3.3.1 The Unconstrained Quadrature Loglikelihood Statistic

A program was written for calculating the loglikelihood ratio appropriate for detecting a signal when the signal, clutter, and noise are described by models in concatenated quadrature form. These models permit the in-phase and quadrature components to be both spatially and temporally correlated.

Let $x(1), x(2), \ldots, x(N)$ be a multichannel return in concatenated quadrature form. That is, each $x(n)$ represents a time sample and is a real vector with an even number of elements. The first half of these elements are the in-phase components, and the second half are the quadrature components. The statistical hypothesis test which the program implements tests between the two hypotheses:

- H_0: $x = c + n$
- H_1: $x = s + c + n$

where s is a signal, c is clutter, and n is white noise. Clutter is optional. A test of this type is a signal detection problem.

To implement this test, the user must use several programs, including a new one written under this effort. The programs for estimating AR parameters and implementing a linear filter are not based on the concatenated quadrature form. They treat the concatenated quadrature signal as a real multichannel signal with twice the appropriate number of channels. To interpret the output of these programs correctly, the user should mentally partition the matrices given for AR coefficients and the covariance matrix. The new program described in this section calculates the appropriate loglikelihood statistic based on the theory of unconstrained quadrature components.

The statistical test is based on an innovations representation. Using the unconstrained quadrature synthesis program, the user can create a file containing a specified number of realizations of the sum of signal, clutter, or noise. Coefficients should already be estimated for AR models of the null and alternate hypotheses. Using these models, the user can pass the synthesized processes through two filters operating in parallel. The output of the null hypothesis filter is an estimate of the innovations under the AR model corresponding to the null hypothesis. The output of the alternate hypothesis filter is an estimate of the innovations of the alternate hypothesis AR model. The outputs of these two filters are stored to files, and can be read by the loglikelihood statistics program.

The loglikelihood statistics program calculates the following statistic [Michels 91]:

$$\ln \Lambda = \frac{1}{2} \sum_{j=1}^{J} \sum_{n=1}^{N} \left\{ \frac{[\varepsilon_{I,j}(n \mid H_0)]^2}{\sigma^2_{I,j}(H_0)} + \frac{[\varepsilon_{Q,j}(n \mid H_0)]^2}{\sigma^2_{Q,j}(H_0)} - \frac{[\varepsilon_{I,j}(n \mid H_1)]^2}{\sigma^2_{I,j}(H_1)} - \frac{[\varepsilon_{Q,j}(n \mid H_1)]^2}{\sigma^2_{Q,j}(H_1)} \right\},$$

where J is the number of channels, N is the number of time samples, $\varepsilon_{A,j}(n \mid H_k)$ is the output of the linear filter corresponding to the hypothesis H_k, and $\sigma^2_{A,j}(H_k)$ is the corresponding standard deviation. This statistic is calculated for each set of realizations in the input. The alternate hypothesis of a signal being present is accepted if the statistic is above some threshold value.

It should be noted that the synthesized returns can be converted to complex form before estimation and filtering. If this is done, the appropriate loglikelihood statistic to calculate is the Gaussian statistic described Section 3.3.2.1. This procedure allows an examination of the robustness of the Gaussian loglikelihood statistic to a model mismatch of correlation between the In-phase and Quadrature components.

3.3.2 The SIRP Loglikelihood Statistic

The SIRP loglikelihood ratio was implemented in the original system for both single and multichannel cases. This section describes the multichannel version.

Given two multichannel filtered processes and variances for each channel, this program calculates certain loglikelihood ratio statistics. Two statistics are calculated, one appropriate for Spherically Invariant Random Processes and the other for Gaussian processes. Many realizations of the returns are input into the program. The program calculates the statistic for each of these realizations. The program operates in two modes. In the first mode, the program allows the user to determine thresholds for a given false alarm rate. In the other mode, the program produces the probability of detection for a given threshold.

3.3.2.1 Background

This program is part of an implementation of a statistical hypothesis test. Let $x(1)$, $x(2)$, ..., $x(N)$ be a multichannel random process. That is, each $x(n)$ represents a time sample and is a vector with each element representing a channel. The test decides between the two hypotheses:

- H_0: $x = c + n$
- H_1: $x = s + c + n$

where s is a signal, c is clutter, and n is white noise. A test of this type is a signal detection problem.

In statistical theory, the *significance level* of a test is defined to be the probability of deciding in favor of the alternative hypothesis H_1 when, in fact, the null H_0 is true. Since for a signal detection problem, this mistaken decision corresponds to erroneously concluding a signal is present, radar theorists refer to the significance level as the *probability of false alarm*. One decides a signal is present when the loglikelihood ratio exceeds a *critical value*. In radar theory, a critical value is known as a *threshold*.

Statisticians define the probability of accepting the alternative hypothesis H_1 when, in fact, it is true as the *power* of a test. Given the physical interpretation of the above signal detection problem, radar theorists refer to the power as the *probability of detection*.

The program described here is designed to analyze specific statistics in terms of false alarm probabilities, probabilities of detection, and thresholds. It is the final module of certain programs that implement specific statistical tests. Given the probability of false alarm, it returns a threshold. Given a threshold, it can be used to calculate the probability of detection.

The statistical tests analyzed by this program are defined here. The models of the signal, clutter, and noise in the above hypotheses may have both intertemporal and cross-channel correlation. The first step in the statistical procedure is to pass the synthesized process through two filters in parallel. These two filters are known as F_0 and F_1.

The filters are designed to remove both the intertemporal and the cross-channel correlation under one or the other hypothesis. If the null hypothesis is true, the output of the F_0 filter is uncorrelated. On the other hand, if the alternative hypothesis is true, the output of the F_1 filter is uncorrelated. Thus, one of the filter outputs is white noise. The system of which this program is a part provides capabilities to generate inputs such that the resulting white noise is either Gaussian or a special type of Spherically Invariant Random Process (SIRP).

For the Gaussian case, the appropriate calculated statistic is

$$\ln \Lambda = \sum_{j=1}^{J} \sum_{n=1}^{N} \left\{ \frac{|\varepsilon_j(n \mid H_0)|^2}{\sigma^2_{j,H_0}} - \frac{|\varepsilon_j(n \mid H_1)|^2}{\sigma^2_{j,H_1}} \right\}$$

where J is the number of channels, N is the number of time samples, σ^2_{j,H_i} is the variance of the jth channel in the output from filter F_i when H_i is true, and $\varepsilon(n \mid H_i)$ is the output from filter F_i.

For the SIRP case, the appropriate statistic is

$$\ln \Lambda = \ln h_{NJ}(q_1) - \ln h_{NJ}(q_0)$$

where

$$h_{NJ}(q_i) = \frac{2\alpha^{NJ/2}}{\Gamma(\alpha)} (\sqrt{\alpha q_i})^{\alpha - NJ/2} K_{NJ/2-\alpha}(2\sqrt{\alpha q_i})$$

$$q_i = \sum_{j=1}^{J} \sum_{n=1}^{N} \frac{|\varepsilon_j(n \mid H_i)|^2}{\sigma^2_{j,H_i}}$$

K is the modified Bessel function of the second kind, and Γ is the Gamma function. α is a shape parameter associated with SIRPs.

Other programs exist in the system for

- Synthesizing realizations of signal, clutter, noise, and their sums
- Estimating filter parameters
- Implementing the filters

The program described here calculates the above loglikelihood ratios, depending on which one is chosen by the user. The statistic is calculated for many trials, and the user can use these realizations to determine the threshold for a given false alarm probability or the detection probability for a given threshold.

3.3.2.2 Detailed Specification

The program prompts the user for the names of two files, one containing the output of the F_0 filter and the other the output of the F_1 filter. These files are binary and should contain an equal number of trials of their respective multichannel signals. They should also contain the same number of channels. These files are in the usual formats for synthesized data for this system.

The program has several options for the variances. The user can decide to estimate means and variances from the data by the usual statistical estimation formulas. Both biased and unbiased options are available. Alternately, the program reads variances from a user-specified file. Finally, the user can enter the variances directly. All these options are independent for the two signals.

The program asks the user whether the SIRP or Gaussian statistic should be calculated. If the SIRP option is desired, the program prompts the user for the value of α.

The program asks the user to choose between two modes, one for calculating thresholds and the other for calculating the probability of detection. If the threshold calculating option is desired, the user is prompted for an index into the sorted threshold list. If the detection probability mode is desired, the user is prompted for a threshold value.

The user is given the option of viewing the statistic on a trial-by-trial basis. Also, the user is prompted to determine if the sorted list of likelihood ratios should be echoed to the user's screen.

The program reads in a pair of filter outputs from the two files and calculates the appropriate statistic. If the user has so indicated, the program outputs the statistic for each trial as it is calculated. The program maintains a sorted list of the statistic's value.

After all trials have been read, the program prints out the sorted list if the user has so indicated. Under the threshold calculation mode, the program outputs all thresholds after the indexed value, inclusive. Under the detection probability mode, the program outputs a count of the number of statistics that exceed the threshold. In either case, the program outputs the total number of trials.

Subroutines were written to calculate modified Bessel functions and the Gamma function under this effort. These subroutines were taken from Kaman's Colorado Springs office, which implemented a modified Bessel function for another project.

3.4 Diagnostics

Rangaswamy et. al. states that a certain quadratic form is a sufficient statistic for SIRPs [Rangaswamy 91]. The distribution of this statistic can act as a good check on the output of the SIRP synthesis routines. Thus, a program was written to determine the distribution of this quadratic form for the white noise driving terms of the AR processes.

The input to this routine is a file containing many realizations of a multichannel radar return. The user is also asked to enter the variances for each channel. For each realization, the program calculates the quadratic form q_j, $j = 1, 2, ..., J$, where J is the number of channels:

$$q_j = \frac{1}{\sigma_j^2} \sum_{n=1}^{N} |x_j(n)|^2.$$

N is the number of times samples, $x(1)$, ..., $x(N)$ is a white noise process, and σ_j^2 is the variance for the jth channel. (For a complex number $z_r + i\, z_i$, the square of its magnitude is given by

$$|z_r + i\, z_i|^2 = z_r^2 + z_i^2.$$

This is a real number.)

The theoretical probability density function (pdf) is not calculated by any program in the system, but is given here for completeness. The pdf for the quadratic form q_j is

$$f(q) = \frac{q^{N-1}}{\Gamma(N)} h_{2N}(q)$$

where

$$h_{2N}(q) = \frac{2\alpha^N}{\Gamma(\alpha)} (\alpha q)^{(\alpha - N)/2} K_{\alpha - N}[2(\alpha q)^{1/2}],$$

Γ is the Gamma function, and K is the modified Bessel function of the second kind.

For each realization of the input process, the program calculates a single real number as defined by the above quadratic form. To ensure compatibility with other programs in the system, these outputs values are stored in a data structure appropriate for a single realization of a multichannel process, where the number of time samples in the output process is equal to the number of realizations of the input process. Each time sample in the output is the

quadratic form for the corresponding trial in the input. This output process is saved to a user-specified file. This file is readable by other programs in the system, particularly the programs for calculating certain statistics and for displaying histograms.

Two related programs were developed under this effort. One calculated the SIRP quadratic form for a single channel signal, and the other calculated data useful in plotting histograms. A histogram plotting program was added to the system under another effort during the time period of this task.

4. ANALYSIS SEQUENCES

The capabilities of the new UFI-based system are implemented by providing the user with certain analysis sequences. Each analysis sequence calls existing modules in a fixed order. The user is prompted for all necessary inputs. The UFI-based system creates and executes a main program using the user's inputs. The outputs of this program provide the user with the desired results.

During this effort, many new analysis sequences were written. In addition, existing sequences were changed to include a SIRP capability. This section describes these additions and changes. The analysis sequences are now organized into four menus:

- A menu of "AR signal sequences" for six previously existing analysis sequences, some of which now allow the user to choose between Gaussian and SIRP noise. The signal is an AR process.
- A menu of three "multipath sequences," in which the signal includes a multipath component.
- A menu of four "deterministic signal sequences." The signal is deterministic, the clutter is an AR process with either Gaussian or SIRP driving noise, and the white noise is either Gaussian or SIRP.
- A menu of seven "unconstrained quadrature sequence," in which the In-phase and Quadrature components of the signal, clutter, and noise are not constrained to be uncorrelated.

4.1 AR Signal Sequences

The sequences for modeling the signal as an AR process are described in greater detail in [Kaman 92a]. Six sequences are provided on this menu:

- AR parameter estimation
- ARMA parameter estimation
- Detection analysis, no clutter, a priori parameters
- Detection analysis in clutter, a priori parameters
- Detection analysis, no clutter, no a priori parameters
- Detection analysis in clutter, no a priori parameters

The sequence for estimating AR parameters generates a number of trials consisting solely of a signal. The signal is described by an AR process in which the driving noise may be either Gaussian or a SIRP. Each trial is used to estimate the AR coefficients and the covariance matrix. The final output is the mean, variance, and error variance of these estimates.

The sequence for estimating ARMA parameters generates a number of trials consisting of the sum of signal and white noise. The signal is generated from an AR process, and both the driving noise and the white noise may be either Gaussian or a SIRP. Each trial is used to estimate the parameters of an AR approximation of the ARMA model for the sum of signal and noise. The final output is the mean and variance of these estimates calculated across all trials.

The final output of the remaining four sequences consists of means and variances of detection probabilities and thresholds for given false alarm rates. The sequence for detection analysis with "a priori" parameters and no clutter is based on an algorithm for detecting a multichannel AR signal in white noise. The noise can be Gaussian or SIRP, and so can the signal driving noise. The null hypothesis filter merely lags the input appropriately and decorrelates it across channels using the actual covariance matrix of the white noise. The alternative hypothesis filter uses coefficients generated by "a priori" parameter estimation.

The other "a priori" detection analysis sequence analyzes signals in AR clutter and white noise. Both the signal and the clutter are described by AR processes with either Gaussian or SIRP driving noise terms. The white noise is also either Gaussian or SIRP. The parameters for both the null and the alternative hypothesis filters are found by "a priori" parameter estimation using a cell' channel for the alternate hypothesis estimates.

The remaining two sequences without "a priori" parameters were not modified under this effort. The AR models used for these sequences can only have Gaussian driving noise terms.

4.2 Multipath Sequences

The sequences written under this effort parallel the estimation and "a priori" detection analysis sequences in the menu for AR signal sequences. The menu for multipath sequences contains three choices:

- Estimate AR parameters for multipath process, no noise
- Estimate AR parameters for multipath process with noise
- Multipath detection analysis in clutter and noise, a priori parameters

The first choice calls a sequence that estimates parameters for a multichannel AR approximation to a multichannel multipath model of the sum of signal and clutter. The clutter is an AR process with either SIRP or Gaussian driving noise. The signal has a multipath term, as described in Section 3.1.3.4. The user specifies the number of trials of the sum of signal and clutter to be generated, and the AR parameters are estimated for each trial. The final output is the mean and variances of these estimates.

The sequence for estimating AR parameters for multipath processes with noise is very similar. The signal has a multipath term, and the clutter is an AR process. The driving noise terms can be either Gaussian or SIRP. The sum of signal, clutter, and noise is synthesized for each trial. AR parameters are estimated for each trial using an algorithm and order selected by the user. The final output is the mean and variance of these parameter estimates.

The multipath detection analysis is the most complicated sequence of these three. The loglikelihood ratio used here is the Gaussian statistic developed under the assumption of no multipath components. Thus, this sequence explores the performance of a signal detection algorithm under a model mismatch. The signal, clutter, and noise are modeled as above. An "a priori" parameter estimation phase estimates the parameters for the null and alternative hypothesis filters. The null hypothesis filter is based on a model of the sum of clutter and noise. The alternate hypothesis models the sum of signal, clutter, and noise, where the signal has a multipath term.

Once the filter coefficients are estimated, a number of trials of clutter plus noise (i.e. the null hypothesis) are synthesized. Each trial is passed in parallel through the filters. The loglikelihood statistic is calculated from the filter outputs. This sample of loglikkelihood ratios is used to estimate the thresholds for given false alarm rates. This process of estimating the

distribution of the loglikelihood statistic is repeated, only based on the synthesis of the sum of signal, clutter, and noise. The detection probability is estimated for the estimated threshold. An outer loop, which includes "a priori" parameter estimation, thresholding, and detection probability estimation supports the calculation of the means and variances of thresholds and detection probabilities. These means and variances are the final outputs of this sequence.

4.3 Deterministic Signal Sequences

The sequences for analyzing deterministic signal models do not include separate sequences for estimating AR parameters. All four sequences are designed to analyze the performance of signal detection algorithms:

- Detection analysis, constant signal in white noise, actual parameters
- Detection analysis, constant signal in white noise, a priori estimates
- Detection analysis, constant signal in AR clutter, actual parameters
- Detection analysis, constant signal in AR clutter, a priori estimates

The first sequence analyzes a signal detection algorithm in which the signal is deterministic, as described in Section 3.1.3.1, and the noise is white, possibly correlated across channels. The noise can be either SIRP or Gaussian. The detection algorithm decides between the two hypotheses

- H_0: The radar returns consist of white noise
- H_1: The radar returns consist of the sum of a deterministic signal and white noise

The null hypothesis filter merely decorrelates the noise across channels. The alternate hypothesis filter first subtracts the (known) deterministic signal and then decorrelates the noise across channels. In both filters a decomposition of the actual noise covariance matrix is used; this sequence does not contain any procedures to estimate the filter parameters. The loglikelihood statistic appropriate for this model is used to decide whether a signal is present or not. The final output consists of means and variances of thresholds and detection probabilities for user-specified false alarm probabilities.

The second sequence differs from the first only in that "a priori" parameter estimates are used for the filters. The user specifies the number of trials used to estimate the parameters for the linear filter decorrelating the noise. This filter actually includes AR parameters for decorrelating across time as well, even though the noise includes no temporal correlation. Hence, the user must specify an AR parameter algorithm and an order for the process of estimating the filter coefficients.

The third sequence is based on a different model. The signal remains deterministic, but the noise is modeled by an AR process which includes both cross-channel and intertemporal correlation. The noise is referred to here as "AR clutter." In this case, the filter for the null hypothesis is a linear filter which decorrelates its inputs across time and space. The alternate hypothesis filter first subtracts the deterministic signal before performing the filtering based on the AR parameters. These filters use the actual parameters of the AR process for the noise, not estimates. As usual, the final output is the means and variances of threshold and detection probability estimates.

The fourth sequence differs from the third in that the filter coefficients are "a priori" estimates; that is, estimates obtained using previously obtained data from a 'reference' channel and a 'test cell' channel.

4.4 Unconstrained Quadrature Sequences

The last menu is for sequences analyzing unconstrained (concatenated) quadrature models of the signal, clutter, and noise. All white noise terms in these sequences are Gaussian; no SIRP options are available. An interesting feature of these sequences is that capabilities are provided for analyzing processes in concatenated quadrature form and by immediately converting them to complex form after synthesis. The latter situation is referred to as a model mismatch, since the theory on which the analysis is based assumes the in-phase and quadrature components are uncorrelated. Seven sequences are available for analyzing unconstrained quadrature models:

- Estimate AR parameters for signal, no clutter, no noise
- Estimate AR parameters for signal + noise, no clutter
- Estimate AR parameters for signal + clutter + noise
- Detection analysis, no clutter, a priori parameters, model mismatch
- Detection analysis, no clutter, a priori parameters
- Detection analysis in clutter, a priori parameters, model mismatch
- Detection analysis in clutter, a priori parameters

The first three sequences analyze various algorithms for estimating AR coefficients. They are parallel in structure. A number of trials are synthesized for the sum of the indicated combination of signal, clutter, and noise. The signal, clutter, and noise are all described by unconstrained quadrature models, as described in Section 3.1.1. For each trial, AR coefficients and the covariance matrix are estimated by a specified algorithm. The resulting estimates should be viewed as partitioned matrices, as is appropriate for the concatenated quadrature form. In parallel, AR parameters are estimated for the complex form of the synthesized return, based on a conversion of the concatenated form. The final output of these sequences are means and variances of both sets of AR estimates.

The remaining four sequences parallel one another in structure. The final outputs are means and variances of thresholds and detection probabilities for user-specified false alarm probabilities. The signal and clutter, which is only present in the last two sequences, are synthesized to fit a multichannel unconstrained quadrature AR model. The white noise is also synthesized under an unconstrained quadrature model. In the model mismatch sequences, the synthesized radar returns are immediately converted to complex form, and the Gaussian loglikelihood statistic described in Section 3.3.2 is eventually calculated. The unconstrained quadrature loglikelihood ratio described in Section 3.3.1 is calculated for those two detection analysis sequences without a model mismatch.

All the filters, both for the null hypothesis and the alternate hypothesis, are based on AR models. Their parameters, the AR coefficients and the decomposition of the covariance matrix, are obtained by an "a priori" estimation process. This description holds even in the cases in which the null hypothesis models the radar returns as white noise. For the model mismatch situation, the actual covariance matrix is not available for the complex form. Hence, an estimation procedure must be performed. Even though the actual value is available for the no-clutter no-model-mismatch case, it seemed advisable to undergo the estimation process anyway so as to maintain the parallelism.

5. REFERENCES

[Dongarra 79] J J. Dongarra, C. B. Moler, J. R. Bunch, and G. W. Stewart, *Linpack Users' Guide*, Society for Industrial and Applied Mathematics, 1979.

[Fishman 73] George S. Fishman, *Concepts and Methods in Discrete Event Digital Simulation*, John Wiley, 1973.

[Kaman 91a] Kaman Sciences Corporation, *Man Machine Interface Experiment*, January 23, 1991.

[Kaman 91b] Kaman Sciences Corporation, *User Front-End Interface Knowledge Base (UFIKB) Builders' Guide*, February 1991.

[Kaman 92a] Kaman Sciences Corporation, *Software User's Manual for the Multichannel Signal Processing System*, May 30, 1992.

[Kaman 92b] Kaman Sciences Corporation, *Enhancement of Multichannel Signal Processing Simulation System*, December 3, 1992.

[Michels 89] James H. Michels, *A Parametric Detection Approach Using Multichannel Processes*, Rome Air Development Center, RADC-TR-89-306, November 1989.

[Michels 90a] James H. Michels, *Synthesis of Multichannel Autoregressive Random Processes and Ergodicity Considerations*, Rome Air Development Center, RADC-TR-90-211, July 1990.

[Michels 90b] James H. Michels, *Multichannel Linear Prediction and Its Association with Triangular Matrix Decomposition*, Rome Air Development Center, RADC-TR-90-226, November 1990.

[Michels 91] James H. Michels, *Multichannel Detection Using the Discrete-Time Model-Based Innovations Approach*, Rome Laboratory, RL-TR-91-269, August 1991.

[Rangaswamy 91] M. Rangaswamy, D. D. Weiner, and A. Ozturk, "Simulation of Correlated Non-Gaussian Interference for Radar Signal Detection," *Proceedings of Twentyfifth Asilomar Conference on Signals and Computers*, Pacific Grove, CA, 1991.

[Rangaswamy 92] M. Rangaswamy, D. D. Weiner, and J. H. Michels, "Innovations Based Detection Algorithm for Correlated Non-Gaussian Random Process," , 1992.

[Robbins 89] Thomas R. Robbins, *UFP Programmer Manual*, Kaman Sciences Corporation, November, 1989.

[Romeu 92] Jorge L. Romeu, *Monte Carlo Validation of A Theoretical Model for the Generation of Non-Gaussian Radar Clutter*, Unpublished technical report, 1992.

Appendix A.
The Single-Channel Minisystem

During this effort, a small version of the original menu-based system was delivered to Dr. Jorge Romeu. This minisystem contains capabilities for single channel synthesis and some diagnostics. It is further described in this appendix.

The system is delivered on a tar tape containing source code and object files for certain library routines. It is intended to be read on a Sun computer. Object code for the libraries is provided for both the Sun 3 and Sun 4 architectures. Reading the tape creates a subdirectory called "schan" under the user's current directory. This subdirectory contains the files described in Table A-1.

To run this system, there must be an environment variable called ARCH already defined in the user's environment. Currently the Sun 3 and Sun 4 architectures are supported and ARCH must be set to one of these two values. Also, the software requires access to the Unix f77 and make programs.

The system is executed by first changing the working directory to "schan." Then "schan [return]" is entered at the Unix prompt to invoke the Single Channel Process Synthesis Menu. The menu contains the following options:

- Generate K-distributed SIRP
- Generate Gaussian noise
- Generate single channel AR process with SIRP or Gaussian driving noise
- Convert a file to ASCII in a format suitable for Dr. Romeu's software
- Convert data to ASCII file

The user should select the appropriate menu prompt. The selected programs are automatically compiled if necessary.

Table A-1: Files in the Minisystem

File	Description
README	Describes how to install and run the system
data	An empty subdirectory
source	A directory of Fortran source code.
source/cbta.f	Converts binary data files to ASCII
source/cgwn.f	Generates single channel Gaussian white noise
source/data2ascii.f	Converts binary files to ASCII in Romeu format
source/scarsirp.f	Generates single channel AR processes
source/sirpm.f	Generates single channel SIRP noise
source/Makefile	Compiles and links above programs
source/sun4.make	Called from Makefile
source/sun3.make	Called from Makefile
schan	A c-shell for invoking the system
hlp	A directory of help files associated with the main programs
hlp/cbta.hlp	
hlp/cgwn.hlp	
hlp/data2ascii.hlp	
hlp/scarsirp.hlp	
hlp/sirpm.hlp	
sun3	Sun 3 object code for various library routines
sun3/libfsl.a	
sun3/libmat.a	
sun3/libmc.a	
sun3/libmv.a	
sun3/libvec.a	
sun4	Sun 4 object code for various library routines
sun4/libfsl.a	
sun4/libmat.a	
sun4/libmc.a	
sun4/libmv.a	
sun4/libvec.a	
bin	A directory to contain linked executables and object code
bin/sun3	A directory to contain compiled Sun 3 object code
bin/sun4	A directory to contain compiled Sun 4 object code
bin/exe	A directory to contain linked executables
bin/menu1	A c-shell for the main menu

MISSION

OF

ROME LABORATORY

Rome Laboratory plans and executes an interdisciplinary program in research, development, test, and technology transition in support of Air Force Command, Control, Communications and Intelligence (C3I) activities for all Air Force platforms. It also executes selected acquisition programs in several areas of expertise. Technical and engineering support within areas of competence is provided to ESC Program Offices (POs) and other ESC elements to perform effective acquisition of C3I systems. In addition, Rome Laboratory's technology supports other AFMC Product Divisions, the Air Force user community, and other DOD and non-DOD agencies. Rome Laboratory maintains technical competence and research programs in areas including, but not limited to, communications, command and control, battle management, intelligence information processing, computational sciences and software producibility, wide area surveillance/sensors, signal processing, solid state sciences, photonics, electromagnetic technology, superconductivity, and electronic reliability/maintainability and testability.